第二节：数据存储之 JDBC

JDBC 核心 API

数据源

接口 - javax.sql.DataSource

获取方式

- 普通对象初始化
  - Spring Bean
  - API 实现
- JNDI 依赖查找

主流 DataSource 实现

- Apache DBCP 1/2
  - 间接依赖 - Apache Commons Pool
    - 对象池的概念
      - “池”化 - “肉少狼多”，“肉”就是资源，“狼”就是“消费者”
        - 特点：有借有还

    - 核心编程思想：生产者/消费者模型
    - 资源：线程资源、数据库资源、I/O 资源
    - 举例：线程池、数据库连接池

- C3P0（字节码提升/优化）
- Alibaba Druid（字节码提升/优化）

JDBC 驱动管理

驱动接口 - java.sql.Driver

驱动管理器接口 - java.sql.DriverManager

管理器角色

获取 Driver 实现

前提：数据库驱动 Driver 实现会显示地调用 java.sql.DriverManager#registerDriver 方法

- 通过 ClassLoader 加载 Drvier 实现（用户/应用控制）
- 通过 Java SPI ServiceLoader 获取 Driver 实现

  加载顺序与 Class Path 的顺序有关系
- 通过 “jdbc.drivers” 系统属性

加载顺序和属性值的顺序有关系

通过读取“jdbc.drivers” 系统属性后，再经过 ":" 的分割，尝试获取多值，再通过 ClassLoader 加载对应的实现类

ServiceLoader 会初始化 Driver 实现类（应用主动配置），包含 Class 加载。

获取 Connection

通过 ClassLoader 类加载数据库 JDBC Driver 实现类的方式，增加 java.sql.DriverManager#registeredDrivers 字段的元素，然后通过迭代的方式逐一 尝试 getConnection 方法参数的 JDBC URL 是否可用。

问题集合

- 当多个 Driver 同时被加载到 ClassLoader 后，到底用了哪个?

getConnection 方法是通过 JDBC URL 判断的，通过迭代多次，返回第一个成功的 Connection 实例

- java.sql.DriverManager#loadInitialDrivers 方法中 Java SPI 空便利的意义在哪里?

```
                try{

while(driversIterator.hasNext()) {
                        driversIterator.next();
                    }
                } catch(Throwable t) {
                // Do nothing
                }
```

`ServiceLoader#next()` 方法会主动触发 ClassLoader 加载。

数据连接接口 - java.sql.Connection

相近语义术语

一个 JDBC Connection 相当于 MyBatis Session 或者 Hibernate Session

创建 SQL 命令 - Statement

SQL 命令接口 - java.sql.Statement

主要类型

- 普通 SQL 命令 - java.sql.Statement
- 预编译 SQL 命令 - java.sql.PreparedStatement
- 存储过程 SQL 命令 - java.sql.CallableStatement

DDL 语句和 DML 语句

- DML 语句： CRUD

  R：java.sql.Statement#executeQuery

  CUD：
  java.sql.Statement#executeUpdate(java.lang.String)
- DDL 语句

  java.sql.Statement#execute(java.lang.String)
  - 成功的话，不需要返回值（返回值 false）
  - 失败的话，SQLException

SQL 执行结果接口 - java.sql.ResultSet

ResultSet 元数据接口 - java.sql.ResultSetMetaData

列的个数、名称以及类型等

SQL 执行异常 - java.sql.SQLException

基本特点

- 几乎所有的 JDBC API 操作都需要 try catch java.sql.SQLException
- java.sql.SQLException 属于检查类型异常，继承 Exception

事务保护点接口 - java.sql.Savepoint

事务

INSERT(S) -> UPDATE(F) -> 回滚

INSERT(S) -> UPDATE(S) -> INSERT(F)

T1 T2

嵌套事务

T0 = T1 + T2

java.sql.Connection#setSavepoint(java.lang.String)

Savepoint t2 = connection.setSavepoint("T2");

t2.

关联技术

Native SQL

数据源（DataSource）

多数据库源（Multiple DataSources）

- N 个 DataSources
- DataSource 代理
  - Druid

MyBatis

MyBatis Generator 通过数据库表结构生成 Java 代码和 SQL Mapper

JPA

相关资料

A -> B -> C -> D -> E

-> Exception -> E

作业

要求

- 通过自研 Web MVC 框架实现（可以自己实现）一个用户注册，forward 到一个成功的页面（JSP 用法）
  - /register
- 通过 Controller -> Service -> Repository 实现（数据库实现）
- （非必须）JDNI 的方式获取数据库源（DataSource），在获取 Connection